UCRL- 95908
PREPRINT

Compcon Tutorial on Computer Security

Russell L. Brand

IEEE Compcon '87 Conference

January 9, 1987

Compcon Tutorial on Computer Security

By
Russell L. Brand

Abstract

This tutorial explains the basic manners of breaking into computers and critiques the various defenses used to protect computers.

*This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

Contents

List of Figures

1 Introduction

This tutorial is designed to be a guide to making a group of computers more secure. It is designed for a system manager and/or system programmer. A familiarity with UNIX is presumed and wherever possible, the examples have been converted to *unix-eese*; though, a few have been left by necessity in terms of VMS and most are footnoted to use VMS terminology, and some comments are included about single user computers.

Later this year, I expect to extend this set of notes into a full book. Comments and suggestions are encouraged and should be sent to

Russell L. Brand
L-306
Lawrence Livermore National Lab
Livermore, CA 94550
brand@lll-crg.arpa
lll-crg!brand

All of the anecdotes in this tutorial are *substantively* true. Company names and individual names have been changed as well as the conversions to *unix-eese*.

There are many explanations that could logically go in any of a number of places, each piece of information is contained in the first section that can easily hold it. To the greatest possible extent, everything is in lists and tables.

In some of the lists a mark in the margin like the one here will appear. ⟸ The ideas in these paragraphs are especially worthy of consideration.

We will consider several class of attackers, defenses, and sites. The three following tables (1, 2, and 3) summarize these classifications.

While there are many categorizations that are useful, this document will primarily use the above.

Crackers. Crackers are people interested in breaking into computer systems. Generally they are not trying to hurt anything, but are just curious or bored and looking to have a good time. Stereotypically, these are children.

Resource Thieves. These are people looking to grab cpu cycles or information, or in some cases, communication access. It may be that some of your users fit into this class.

Spies. Governmental or industrial. They may be interested in stealing data, or in just damaging your system. Again your own users may be on this list.

Figure 1: Classes of Attackers

Administration. Policy decisions.

System Manager. Altering initialization and configuration files.

System Programmer. Rewriting system code.

Figure 2: Source of Defenses

Schools. (machines for class use and machines for administration)

Companies.

Governmental.

Figure 3: Classes of Sites

Part I
Attackers and Attacks

2 Crackers

We will begin by looking at Crackers. Generally we find that they are school aged with lots of time on their hands.

The Cracker can be imagined as someone that would dial up your computer and guess at passwords until successful.

Imagine if you will, a person that is perfectly willing to spend all night doing this entirely by hand. Then realize that

1. His computer will do all of the above while he is asleep, at school or doing something else (perhaps on the same computer).

2. He didn't have to write a program to do it.

3. If he gets into a *good* computer, he will win prestige.

4. His program has built in some better-than-average means of guessing passwords (this will be discussed at some length later).

5. The phone calls are free (*i.e.*, stolen)

Name: Cracker.

Goal: Fun. Amusement. Prestige within his community. Games. Free Telecommunication Services.

Special Skills: Patience. Free Access to Telecommunication Services, (sometimes including the ability to trick the Telecommunication Carrier). Belief in immunity to prosecution.

Favorite Attack: Guessing Passwords. Tossing lots of control characters at canned programs. Looking at discarded sheets of paper.

Figure 4: The Cracker

6. Often he is under 18 he doesn't think that there is any chance that he will end up in much trouble doing this even if caught.

Immediately the mind races to two questions. First, what will he do (or perhaps what can he do) once he has gotten logged on? And second, how can we stop him? We will begin with the second question first. There are some classical approaches to this.

2.1 Classical Ways to Stop a Cracker

A large number of strategies to discourage Crackers have been formed. Many of these provide hassles for the legitimate users of the system and hence are activately fought. Below we survey a variety of strategies that have been used.

Hang Up on 3. This is the strategy where you hang up the phone after three bad login password attempts. You can alter this to hang up on just one bad attempt and find that many of your users get frustrated.

```
Do Until Bored
Step 1. Select and dial a telephone number.
Step 2. If the telephone number has a modem tone on it, proceed. Otherwise, return to step 1.
Step 3. Guess a username and password.
Step 4. If this gets you onto the system, start playing with the machine, otherwise proceed.
Step 5. If you have not guessed too many times for one phone call and the phone has not been hung up, return to step 3, otherwise proceed.
Step 6. Redial the same number.
Step 7. Go to step 3.
```

Figure 5: A Simple Hacker Attack

Alternately you can choose a higher number with less frustration and greater risks. The idea here is that the cracker will have to redial the phone and that he will (1) not want to do this and (2) that it will slow him down a lot. This idea was once worth a little bit; but, not much even then, as the dedicated cracker will (1) happily redial by hand hundreds and hundreds of times or (2) his computer will detect the hang up and redial for him.

There is a generalization of this fallacy where some people believe that you don't have to start logging bad password attempts until the nth bad attempt from a single phone call. For Unix machines, in the default configuration this would be five attempts. Of course your clever attacker just guesses four times, hangs up and guesses four more times for as long as he wants.

This same type of procedure has exactly the same problems when one tries to use it on network connections.

Don't Hang Up But Stop Listening. This is a variation of the strategy of hanging up after 3. Instead of hanging up, we can just continue to show login prompts and such until the attacker stops. Of course we ⇐ only listen to his first attempt of three. Since many crackers are not using programs that are smart enough to hang up on their own, they might continue to guess in vain all night. The intelligent cracker will not fall for this.

It is *very important* to warn the legitimate community about systems that start ignoring things so that the legitimate user does not end up wasting a lot of time in misdiagnosing a problem. If the cracker knows that your system is doing this, this defense becomes almost useless.

In terms of user education, there is a lesson we can learn from the banking community when ATM cards/machines were very new.

> **Once upon a time there was a student with an ATM card. It had a four digit password that had been selected in a perhaps reasonable way. He knew the password had two 1's and two 3's in it. We quickly see that there were only six combinations.**

- 1133
- 1313
- 1331
- 3113
- 3311 (this was the right combination.)
- 3131

This would seem to be a simple enough problem. You just try six times and you get your money and go into the city for dinner. Unfortunately, our young friend didn't read his instruction manual; though even if he had, he would not have found out that the machine *eats* the card after three unsuccessful attempts. Alas, he ended up eating in the cafeteria that night instead.

If our young friend would have been a computer hacker[1] rather than mathematician, he might have known to try only one attempt and then remove the card before trying a second one as this particular type of machine didn't (and still doesn't) keep track after the CANCEL key is hit. This is of course just like the Cracker hanging up the phone and re-dialing.

Invalidate Accounts. Here we turn accounts off when there have been too many bad guesses against them. You could turn them off for a little while or forever. You might turn the account off for exponentially increasing amounts of time with more bad guesses. There are at least two problems with this; the first is that a cracker can easily harass users, as illustrated below:

Once upon a time there was a university that allowed forty bad password attempts per account, per term. Once the forty were used up, one would have to go to the system manager and show some I.D. and have the account reset. The system manager would get a list of all the bad password attempts,

[1] Hacker, not necessarily Cracker. A hacker is someone that is dedicated and good with computers and has a great deal of esoteric knowledge while a cracker is one that breaks into computers

that is, the actual strings that were typed, so that she could notice things like sequential guessing.

Well, it so happened that the system manager was a very prim and proper person that was easily offended. It also happened that there was a student that didn't like this system manager and he decided to invalidate the accounts of all of the prim and proper students, selecting choice pieces of profanity as attempts, and managing to (1) make the system manager blush a lot and (2) getting lots of cpu time for himself by getting rid of many of the users for a day or two at a time, especially over weekends while the system manager was gone.

Second, the Cracker might keep the system manager from logging in once he gains entry, by giving the wrong password for each of the system manager accounts.

Some programs only keep count when guesses are made consecutively one the same account. This allows the attacker to guess a single password and try it against each account with the system ringing the alarms. Other wrong ways to implement this allow for the system to be tricked in similar manners.

Invalidate Phone Lines Here rather than try to disable the accounts, we could disable the contact service. That is, if we are being attacked by phone line, we discontinue accepting logins from the phone lines and allow the logins from hard-wired lines and the network; if we are being attacked from a given hard-wired line, we could disable that particular line. The obvious downside of this is that your Cracker can potentially deny service to all of your people at home, even if he can't deny service to everyone. ⟸

You can generalize this and disable service on one type of modem or to all but certain accounts (that are known to have very good passwords) when the phone lines are being attacked.

Call Back. Here you let our computer dial the user up. In some manner, the user identifies himself and the computer calls the user back at the user's home. In theory this would be very powerful because either (1) ⟸

you would limit the number of places the computer could call or (2) you would have the telephone number of your attacker. Unfortunately, there are several ways for the cracker to defeat this.

Not hanging up. Some modems don't notice that the line has not been hung up when the carrier tone is discontinued and will then hang up their end of the phone. In many areas, the line is not disconnected until the party that initiated the phone call hangs up, and hence the modem will send dial tones onto the line that is still connected rather than succeeding in dialing a different phone number.

Fast Redial. While the modem is preparing to dial out, the attacker can dial in. If the attacker times it correctly, the modem will pick up the phone and answer this second call thinking that it has picked up the call to dial out.

Call Forwarding. It is sometimes possible for the cracker to reprogram the legitimate receiving number to forward to the crackers phone.

Doing *call back* correctly is harder than it sounds. Most often, call back is done with the call the computer makes going out on the same line as the call that came in requesting it. The problem is that it is often very easy to call the computer while it is about to make an outgoing call and let it think that it has called who it wants to call, when, in fact, the same attacker has just dialed in a second time. Further, in some cases, it is easy to make a computer believe that it has hung up the phone when in fact it has not succeeded in doing so.

To do call back correctly you must use a different line to call out than the one that you have used to receive the call. Ideally this should be a *dial out only line* that is not equipped to take in coming calls. Just setting the modem not answer the phone is not good enough.

If the line is not an out *dial out only line*, you should protect this phone number. No One needs to know what it is. While this is not impossible for the attacker to find it, it is in fact very difficult.

Unlisted Number. You could hide the telephone number. If you don't publish the telephone number for your computer, the crackers will have to do some work to find it. Dialing up every single number in an exchange and listening for dial tones is a good way for your cracker to do this. As a matter of fact, he often does.

No Carrier Tone. To prevent the cracker from scanning, you might have your computer not answer its phones with a carrier tone but rather with something else like a tape, or human speech. The user would ⇐
then know to type a password in from his touch tone pad or supply a non-standard tone so that the computer would know to supply the initial carrier tone.

In all cases this approach is inconvenient for your legitimate users (and would require significant work in most cases to get programs that use phone lines to continue to work (*e.g.*, uucp)) and if widely adopted, the crackers would regard this as another level of password. If in fact one can hide the telephone numbers of the computer, this ***would*** increase the work effort that the cracker needed to go through by a factor of

$$\frac{\text{number of phone lines in your company that answer}}{\text{number of phone lines that have computers on them}}$$

Of course you must get a lot of your phones to answer incoming calls before this does you much good.

Expensive Modems. Several years ago the cost of a 1200 baud modem, using the Bell 212a standard, was enough to discourage some of the crackers. To attack a machine that had only 1200 baud input lines, one would either need one of these very expensive modem or to attack through a machine that had both these and modem and ones that would talk to cheaper types the attackers had. The cost of these modems is no longer a deterrent.

Once upon a time, there was a mythical university that wanted to discourage crackers from dialing into their machines. Their staff members wanted to be able to work from home so they needed to have modems. They decided to by modems that used the Vadiac 3400 protocol instead of the Bell 212a protocol. At that time the cheaper modem that would use the 3400 protocol was a thousand dollars, and while they spent an extra $20,000.00 to by these it was enough to prevent crackers from attacking them by phone line. And they live happily ever after (until a local area net connected them to the educational machines).

The obvious drawback of this is the cost. If you have many employees with modems you might alter your cost from under $100 per modem to over $500 per modem.

Encryption. If you do transport level encryption[2], the login banner will look like line noise (or TECO[3]) to your attacker. Further all of his guesses will look like line noise to your computer. Of course, no one will actually think it is line noise or TECO, but it is about as intelligible. If all of your dial-ins will be from people using small computers, encryption *may* be a valid choice for you. If your users aren't all using personal computers, hardware boxes to do the encryption may provide an acceptable answer. If you are running a government installation there are legal restrictions as to what you may do with cryptography.

The problems with password management are different/easier here then they normally are; in fact, much easier for a moderate size user community. You can simply give everyone of your users a disk with the the encrypting terminal protocol, and have no one ever type a password. Every year or so you could distribute new disks. In much the same way you could have a five thousand character password that is resident on such a disk that the terminal protocol transmits rather than using encryption. It makes your communication almost as secure as the physical media. This protection scheme requires that you protect these disks.

[2]Transport level encryption encrypts *everything* that is sent over that line.

[3]TECO is a text editor which has been used to write large systems of macros. It is legendary along with APL for its unreadability.

Nets Instead of Phone Lines. Here you seek to avoid the entire phone line attack problem by using a network instead of dial-ups. The obvious question is whether the network is harder to get into. For the case of commercial nets that are designed to provide convenience and smaller telecommunication bills rather than security, the answer is "No."

Limited Dial Locations. Sometimes it is possible to arrange that only certain phones can dial into the phones that your computers are using. This is the case when one has a private switchboard. For the rare occasion that this is possible, and that your local switching system is not programmable from the outside, this may work.

No Phone Line/No Net. This tends to work very well, especially if you have big barbed wire fences around your facility. It prevents most external attacks, leaving only psionic ones and physical infiltration. Further, by preventing many of your employees from working at home (unless you provide them with PC's and floppy disks and your work makes this practical) you might improve the health and personal lives of your employees. Alternately, you may just be guaranteeing that your computer gets wasted 16 hours a day when everyone is gone.

Better Passwords. Using better passwords makes it harder to guess passwords. Section 7 is devoted to passwords.

Biometrics. In addition to the password itself, there are other things that can be measured about a user when he is identifying himself. For ⟸ people, we have mechanisms such as pictures and thumb prints. For voice communication, we have voice prints and the sound of the voice. In some cases, we ask personal questions (like the infamous mother's maiden name.)[4]

Over a telephone line we can, in some circumstances, get information about the relative pauses between key strokes.[5] This proves to be

[4]Not less than three banks have asked me for my mother's maiden name as part of identifying me over the phone. This would not surprise me except that I never gave any of them her maiden name to begin with...

[5]The interested reader is referred to John David Garcia's paper in the proceedings of this conference.

a viable mechanism for identification when an unbuffered key stroke stream is available.

Dull Headers. Does your system look worth breaking into? If it does, it shouldn't. Does your system encourage non-users to try to log on? Wording, such as the examples below, which appear before logging on is probably a mistake.

```
Welcome to . . .
Brand-X Secret  Computer
. . . Classified . . .
. . . Unclassified . . .
. . . Weapons . . .
. . . Financial Records . . .
. . . Games . .
To login as GUEST . . .
For help . . .
```

There are in fact significant legal implications to using the word Welcome in your login banner. Further, telling the not yet logged in Cracker that it is Unclassified tells him that you are likely to have other interesting computers near by; perhaps even networked to this one.[6] In short having dull headers is a good idea. ⟸

> Once upon a time there was a mythical bank that had an at home banking system. Customers could dial up access to their accounts and do a variety of interesting and useful things. A cracker logged into the system and did a great deal of damage. He was caught and brought to trial.
>
> The judge determine that the pre-login banner saying "Welcome to . . ." encouraged him to try to use the system even though he didn't have an account and therefore found him entirely blameless.

[6]It is in general both illegal and unsafe to have a connection from a *classified* computer to a computer with access to phone lines or unclassified networks. We will talk more about daisy chaining later on.

Dull Front Ends. We could go a step past the dull login message and have an entirely dull front end so that the computer would even look dull if the Cracker broke in. In practice, this is somewhere between hard and very hard.

In general what it takes to stop the Cracker from getting into your system is being more careful than he is clever. Being careful that

- A person needs to know a Legitimate password to get on. ⟸
- It is hard to steal or guess a password. ⟸

The cracker has also been known to have a great deal of success by simply typing `^c` at a number or programs rather than supplying them with the passwords that they request. Many programs have just the wrong failure mode.

2.2 What the Cracker will do if he gets on

Since we know something about our crackers goals we can easily understand what he might do once he gains access.

Privileges. He will try to get as many privileges as he can. This makes any other goal that he has easier to accomplish. It also enhances his prestige. He will also seek to make his access permanent if he can.

Games. and/or neat things to read.

Spread the Word. To get external recognition for something, he must tell someone. Generally this includes enough information so that they can get in as well.

One small step for man. . . He will want to use your computer to get onto other computers on the same net.

Essentially he has a goal of having fun. Don't let this discount the cracker as a threat. Please realize that a cracker can evolve into a thief. ⟸ In fact, by making it easy, anyone can become a thief.[7]

[7]Except me of course.

Once upon a time, I walked into a mythical terminal room and noticed that I had stopped hearing key clicks. I thought the computer might have crashed and expected to hear some cursing. I then overheard a cracker talking to himself:

> "Well, I really shouldn't have broken into this machine. I feel very guilty about it. . . but, as long as I'm here, I might as well steal something."

Then I heard the key clicks begin again and I knew I could go back to my work.

2.3 The college Cracker

The college Cracker is a very different entity. He is not willing to waste a lot of his own time. He doesn't in general have a lot of patience; but instead has a fair amount of computing power and programming ability. He also has a very strong preference for the use of computer networks, especially the *internet*, which is for clever hacks rather than lots of password guessings. He also tends to like bigger computers.

College Crackers are not always college students. Often they are just employees. Perhaps this is an appropriate classification of people that want information, or more accurately are wiling to provide information say as

> Sure, I can break into the registrar's computer to get the number of that cute blonde in the third row. . .

We can pretty much divide the College Crackers into three groups based on their goals.

Crackers Much like the ordinary crackers goals in figure 4.

Hackers. They generally want to get something (perhaps their job) done. They will not let *anything* stop them. This includes the machine they are supposed to use: being too slow, not having enough disk space, being down or not having the right software. These are the people that will break into machines to steal the bug fixes that the system administrators have not yet had time to install. Yes, this is time for another story.

> Once upon a time, in a mythical government laboratory, there was a programmer who was very fond of an editor called EMACS[8]. Unfortunately for our young friend, his favorite editor was not available yet and the Free version of EMACS for Unix (called GNUMACS) was not yet a reality.
>
> Our young friend had deadlines to meet and wanted to have the best tools possible in order to do his job well and finish on time. Obtaining a version of EMACS was relatively easy but installing it required root Privileges on his machine.
>
> Of course he got help to break into the machine (using the `/usr/lib/crontab` bug discussed in section 15), installed his editor, completed his work and eventually got saddled with computer security responsibilities for a lot of machines and lived happily ever after.

In general Hackers that are willing to be Crackers when they need to be are very useful and should be encouraged to be helpful. Often they can be supervised to do everything except actual installations without Privileges and then a trusted person can do the installations. Often these Hackers can be trusted and can provide you with better, faster and more reliable systems.

Among the dangers of Hacker/Crackers is that of competence. It is sometimes the case that they will do serious damage just trying to be helpful. In fact there are many people out there that know just enough about computers to be a danger to themselves and others.

Cheaters. Some of the goals here are obvious. Stealing exams. Changing grades. Often changing other people's grades. Less obvious is a destruction goal. If he can't change the grade he wants, he might destroy the whole computer or database hoping that it can't be recovered. Similarly he might crash a computer to get an extension on an overdue assignment.

[8]EMACS is a screen editor written by Richard M. Stallman at MIT. Great religious wars have been fought over editors and those surrounding EMACS and Stallman's views on the ethics of software could fill a book in themselves

> In industry you find similar behaviors not only in embezzlers but also in employees that want to read or change their performance reviews.

There are a few attacks that our College Cracker is especially fond of that are worth discussing.

Take the Password File Home. In many operating systems it is possible to get a copy of the encrypted passwords. Bringing these to a convenient place and guessing in a safe environment is easy. (And easily prevented.)

.Rhosts. Daisy chaining accounts by proceeding from one account to another with `.rhosts` or `hosts.eqv` is very common. Essentially these facilities allow one to log on to another machine as a trusted person from the first machine without supplying an additional password.

Paths. Noticing when the paths are wrong so that he can put a program in the "right place." In general if he can place a program in a place where system binaries are stored, he will get the user and/or operator to use his program instead of the right one. `/usr/local` is often a good place to try, the library of editor macro's is another.

Classical Trojan Horses. This is the program that looks like a login shell, very good for public terminal rooms.

Be Useful. If he can write a program people want, they will find a way to use it. Even if you audit the first version, does the subsequent version with bug fixes ever get audited?

Drop Class. A simple strategy, an analogous one of dropping projects works just as well.

> Once upon a time, back when computers were expensive and machine time was hard to get, there was a student that wanted to do really well in his computer science class. From the oral tradition he knew that the biggest problem was insufficient computer time. For all the assignments each class

Name: College Cracker.

Goal: Fun. Amusement.

Prestige within his community.

Games.

Usable Software.

System privileges and resources.

Wandering networks for the sake of wander networks.

Meeting neat hackers.

If it is a machine that he is also legitimately using, to fix what he sees as problems, to install better software.

Exams. Homework Assignments. Grades.

Special Skills: Belief in not being overly punished even if caught.

Access to source code.

Team work.

Good technical understanding.

Legitimate access to a variety of networks.

Favorite Attack: Stealing password files.

Computer network holes.

Published bugs.

Bugs apparent by reading very grungy code.

Guessing Passwords.

Tossing lots of control characters at canned programs.

Figure 6: The College Cracker

got only $75.00 of *virtual* money. This was twenty hours for the ten week term.

Being very clever, he registered for three computer classes and hence got three accounts with a total of $225.00 virtual dollars. He did none of the assignments for the two classes he had no interest in but used the the first five weeks (until drop date) of time to work on the assignments of the class he was planning to take, using the money from the other two classes. . . he dropped the other two classes and hence had the full $75.00 that his real account began with to do the second half of the terms work.

"Help" New Students When helping a new professor or student to set up an account, it is very easy for the college cracker to Trojan horse an account. Similarly, the helpful Cracker might ask for a password in order to fix something for someone.

Paging Files Rummaging through them is fun. Amazing what you can find. And it's fun!

Network Bugs. The average college cracker has heard of all the user level network bugs and pre-installed Trojan horses. He can even write programs to use the non-user level ones.

3 Resource Thieves

In this section, after a brief look at who the resource thief is, we will begin our discussion about resources to be stolen other than information and tangible goods. Then we will talk about information theft. We will conclude the section with discussion of using the computer to steal other resources that you may have.

We should note that there is a special type of resource thief that we might call the *wasted cycles thief.* In general he is looking to grab the computer for nights and weekends when it isn't being fully used. In the case of your employees who want to use the computers off hours, it might be best to just allow them to do this. There are a number of hazards that must be considered with this however.

1. Your user may damage something accidently. This hazard can be minimized by proper set up of your machine.

2. The project or methodology that your user desires to undertake may be prohibited by law. An example of this might be for private profit use of government computers. Logically, we would say that if the machine is not being otherwise used, it is better to have it put to some use than for it to lie idle; but, there are often legal restriction as to the use of the computer.

3. The employee might get help to work on his project. This other person might in fact be a *spy* or sufficiently incompetent to be dangerous to your machine.

   In the case of an academic project he might allow his classmates access to the machine.

4. The employee might set up unauthorized modem connections to the machine so that the machine can be used from home. In fact this is a more general problem–especially if you don't know about the configuration of the added entryway he has set up.

3.1 Computer Resources

Imagine if you will, a man with an idea. A truly wonderful idea. An idea that will make him a fortune. If only he had a giant computer to run the simulations. Fortunately he has yours. You might ask yourself whether your accounting practices are good enough to even notice.

Alternately he might be in need of disk space. After all, the only thing truly portable between all computers is the message:

Disk Space is Tight. Please remove unneeded files Now.

He might even be in another division of your own company, or in fact just on your same net. Are there any nice, big, publically writable areas where he can store his files?

Name: Resource Thief.

Goal: Machine Cycles.

Information or programs.

Special Skills: Profit in mind.

Potential access to your machine room.

Possibly your trust.

Favorite Attack: Bribery.

Figure 7: The Resource Thief

3.2 Information

Our friend above might just as well want some of your proprietary programs. After all it won't hurt you any for a copy to be taken. In fact you probably wouldn't notice. Of course, his being on a competing company payroll and having the program or database free when you have paid hundreds and thousands of dollars for it might give him a competitive edge.

We will defer most the discussion on the theft of the data that your company has created until the section 4, and here will make a short point about databases.

> Once upon a time, there was a company that had two seemingly conflicting goals. You see they had a great deal of data about individuals, in fact, they had employment data on tens of thousands of them. While they very much wanted to protect the information about each individual, they also wanted to make the aggregate data available to statisticians and researchers.
>
> Using a commercial database system, they restricted the

> queries for the researchers so that no query could successfully be made that spoke of less than 150 people.
>
> Through the use of very clever queries (simplified versions of them shown in figure 8) they were able to get the exact salary of every employee in the company.
>
> While I admired the mathematics and inventiveness of these attackers, I couldn't help but giggle when I found that all the data files were publically readable on the system and all they had really needed to do was to type `cat` to get at them.

In general, care must be taken to protect the actual data files as well as their access through the standard programs. this is ***especially true*** when the data files are not stored carefully, on floppy disks for example. It is also the case that just providing aggregate data doesn't tend to restrict information as strongly as our intuition would tell us that it does.

3.3 Computer as a tool

As in everywhere else in this document, it is useful to realize there are two different levels of attack. First there is the user level attack which is often well exemplified by the cracker's methodology discussed previously and our programmer's methodology, similarly shown in section 2.3.

It is generally the case that systems are vulnerable to attack at user level. The thief doesn't need to be a programmer to steal. Let us consider ⟸
a fairly typical scenario.

> Once upon a time there was a keyboardist. She worked all day at a major financial institution entering fund transfers from written sheets. She didn't find this work terribly rewarding.
>
> One day, at the close of a quarter, she saw that things were very busy. Everyone had to work overtime. The computer was so loaded that they turned the accounting department software off to squeeze every last computron out of the machine. The keyboardist found out that every quarter was like that. She was so stressed that she began to get ulcers.

```
==> What is the average salary of the sales department

$25,032.37

==> How many people are in the sales department

 1068

=> How many left handed people are in the sale department

 1

=> What is the average salary of the left handed people
in the sales department

 *ACCESS VIOLATION*

=> What is the average salary of the right handed people
in the sales department

24,987.30

=> Comment.  The math is left as an exercise to the reader.
```

Figure 8: A Sample Database Attack

> Finally it dawned on her that she could put her account number into the right place on the form, and make her account the recipient of many of the large transfers.
>
> It required no programming on her part, only typing the "wrong" field into the computer a couple dozen times. By the time anything had been discovered by the company, she was long gone.[9]

Whenever information flows from one place to another, we must be careful that it arrives intact[10]. The weakest link in this process is human transcription since (1) humans are "noisier" (more prone to error) than almost any other communication media and (2) humans are a bit more "creative" than any other communication media.

What can actually be done to prevent situations like the one above? First and most obvious, they should not have turned off the accounting software. Perhaps with all the records they could have figured out where the money had gone.

Second, having a single human being type in the data is both prone to human error and theft. Having two separate people type in each transaction would have been more expensive in the short run but much better in the long run. In fact, it is possible that the savings in preventing honest mistakes might have made it cost effective to have redundancy here even without the consideration of theft.

Third, most user level embezzling attacks are based on honest mistakes that were not caught quickly. After all, if it wasn't caught as an accident, who would think that it would be caught if done purposefully?

> Once upon a time, there was a mythical investment house. It had an interesting error handling policy. Anything that resulted in an access violation, including the user (in this case the stock broker) mis-typing his name, or a clients account number, resulted in a phone call. Generally within five minutes.

[9]Yes, there are a few details left as an exercise for the reader.

[10]In many cases we want to also be sure that no other copies have been made. This is very hard

> While the brokers were at first very frustrated with this, and offended a few clients by cursing under their breaths too loudly, they eventually accepted it; many taking touch classes as a result.
> Beyond the actual physical protection it gave them, the understanding that the management was ready to even do things like this, was enough to strongly discourage theft. The company saved an estimated $800,000 during the *second* year from the reduction in the error rate.

We should of course realize that there is a point of diminishing returns. ⟸ It is not worth spending tens of millions to prevent loss in the tens of thousands.

---

1. *Never* Turn off accounting.
2. Require two people to independently authorize important transactions.
3. Notice all errors *quickly*. Including honest mistakes.

Figure 9: Embezzlement Prevention

Beyond stealing money, the computer can be used to steal goods and service in many interesting ways.

> Once upon a time, in the old days of mail order, there was a man that made a great deal of money by sticking new mailing labels over the old ones that had the legitimate address on them. It was easy, relatively safe and required no heavy lifting. His much lazier son, who eventually got a job in the same company, improved upon his father's scheme by having the computer misprint the mailing labels. He even used the same post office box.

Free services are in many ways easier to arrange than free physical items. It is somewhat harder to count services carefully. If your computer controls the counting, it is a good target. A physical counter of some type that you can spot check against the computer count is a cheap way to buy some peace of mind. ⟸

4 Spies

Spies are a fun group of people. In general they have lots of resources and very little concern with prosecution. We will consider corporate spying as well as government spying. There is a great deal of overlap between the spies and the resource thieves though the spies can have an added goal of destruction.

Name: Spy.

Goal: Information.

Special Skills: Patience. Lock picking. Bribery. He may be one of your employees.

Favorite Attack: Media theft.

Figure 10: The Spy

Presumably, if you are a government site or contractor dealing with classified materials, you already know how to keep them safe. As a result, this discussion is targeted at corporations that have data they wish to protect.

Once upon a time, in a mythical classified laboratory, a young cracker came in for an interview. As he was led through the halls by a trained escort, his eyes glanced into every open office.

Scheduling such interviews was always difficult, and he found himself with his escort in a secretarial ante-office while waiting to see the Big Boss. While waiting, he saw taped to the secretary's terminal the username and password of the Big Boss. Also the computer's telephone number.

While there was no classified information on the computer, he was able to send the correct computer mail messages to have himself hired. *Unfortunately*, the password was changed just before he could give himself a raise.

4.1 Theft

The most common attack is media theft. This can be accomplished by just going into the machine room and stealing a backup tape. Lock your machine room/tape vault well. Understand who can get in. Don't forget about the *invisible men*, like janitors.

If you concerned about this level of attack, the machines with data probably should *not* be on networks or on phone lines. Physical security $\Longleftarrow$ is the primary tool.

You might consider encrypting your backup tapes. If you are using encryption, this is not the time to do amateur cryptography. Using the crypt facilities that come with your operating system is amateur cryptography. They are *often* very poor encryption methods specifically designed to allow the files to be regained when the naive user has forgotten the key.

A good way to lose data is to not completely erase your media. Thinking that a tape you don't need any more can be re-used or discarded is a dangerous pre-supposition. There may be sensitive data on it that will not be overwritten because the new data set is smaller. Sophisticated attackers can get back several previous writes from your magnetic disk or tape.

4.2 Destruction

The first thing to realize is that you are almost guaranteed to lose against someone that is truly dedicated to destruction. Your goal is to prevent the computer from being the weakest link in your defense; no matter how good your hardware and software and backup schemes are, it is very unlikely to be a match for a couple of well disguised exploding Coke bottles

Denial of service is a weak form of destruction. It is easy to crash a computer. This prevents people from getting any work done on it. It is also relatively easy to crash a computer so that it will be very hard to bring it back up.

> Once upon a time, there was a mythical computer break-in contest. In it the contestants were asked to read or modify a file named `/hidden/Unicorn` and a prize was offered for the first person to do so.
>
> While no one was able to read the file, one clever contestant was able to destroy the file. He made lots and lots of links to the parent directory. Well it seems that the link counter was "wrappable." That is to say that one could keep incrementing it until it didn't know how to store such a big number so instead stored zero, like the speedometer of a car wrapping around at 100,000 miles.
>
> The directory had a use count of zero and was garbage collected by the system. He then generated enough disk activity to make sure the file was not salvageable.

It is somewhat harder to break the physical hardware but not impossible.

> Once upon a time at an early computer science lab, a young "urchin" came in and claimed to be able to break a disk drive with software. The system programmer said that perhaps he could clear the disk or corrupt the data, but one simply could not break a disk drive with software.
>
> The "urchin" typed in his assembly language program that did the disk seeks in just the wrong order. Slowly they could see the disk drive wobble and suddenly fall.

Of course, it is easier to deny service just by tying up the machine and slowing it down a lot. This is often very easy by simply making several programs that use memory in very dumb ways. Accessing every element in two dimensional array in the ascending order along the wrong dimension is a common way to do this.

Redundancy, as mentioned earlier, is a way to assure that things you want will be there when you want them. Keep a second copy of anything that is important. Hardware. Software. Files. Tapes. Manuals. ⟸

Frequent Back Ups. Discussed in length in section 6

Spare Parts. Keeping spares in your machine room of the parts that most often fail is a good idea.

Multiple Copies on Disk. Just keeping a second copy of the important files on line helps somewhat. It helps protect from accidental deletion ⟸ and corruption. Keeping them in separate directories and even on separate disk drives is even better.

Mirrored Disks. Doing every disk write to a second disk as well.

Mirrored Computers. Doing everything on a second computer as well or having a second computer as a spare.

4.3 Computer For Entry

What the spy may wish to steal may have absolutely nothing to do with your computer. There are many places where the computer is used to help identify people. To tell us which key cards can get into which rooms and such. If the attacker can get into the computer, he can authorize a number on his own key card. Further, the database on which cards are allowed access to which rooms must be protected so he can't make a card to match the data that is there.

If you are using the computers to control entry in this fashion there are several things you should check.

- Take a blank card that isn't in your database yet and see if it can open anything.
- Place some cryptographic information in the card, otherwise it is very easy to make cards and master cards.

- Remove old employees from the database even if they return their cards. It is not that hard to make a copy.
- Take a keycard and put it in a microwave oven. Then try to use it. In at least one case, the type of pattern that this left would work; it essentially left all "ones" on the magnetic strip.

Similar precautions might be considered for copy cards and other cards used to provide people with items and services, as faking them takes exactly the same type of knowledge.

5 The Insider Threat

The first thing to realize about the insider threat is that any person can become an insider for the cost of bribing one employee. As discussed in section 3.3 we spoke about how honest errors that are unnoticed make it tempting for someone to do more explicit error creation.

The second thing to realize is that an insider is free to use any of the outsider attacks!

There are a number of things that you can do to make it easier for one of your employees to become a threat. You can make it easier for him to decide to do it. You can make it easier for him to get away with it. Below is a list of *my* favorite ways to make crime more attractive.

Don't Notice Errors. If you don't notice innocent mistakes, and let employees know that you have noticed them, you are inviting them to cheat you. ⟸

Bad Locks. Locks that are so easy to bypass that carrying a key is superfluous. No one respects a policy of locked doors when they know it is going to take them longer to get into their own machine room with a key than it would for a thief to get in. *It is important for people to believe that physical security is providing them with some real security for the extra effort they are expending.*

Bad installation of good locks is as much of a problem.

> Once upon a time in a mythical company working with state secrets, every lock used was certified as good by their security agency. Unfortunately, the door jams were so warped that each lock could be opened with a pocket knife.

No Locks. If the machine room door is unlocked, one is much more likely to go in.

Identical Combinations. On lots of locks.

> Once upon a time there was a mythical laboratory that partially had a good lock policy. Every computer room and every terminal room was protected by a combination lock. Further, they used the best possible, most difficult to pick locks. However, every lock had the same (very easy to guess combination) which was posted in the company cafeteria.

Unprotected backups. As mentioned before.

Open Consoles. If I can get to the machine console, I can most assuredly get into the machine.

Open Machine Rooms. If I can get into the machine room, I can get into the machine.

Stealable Console Logs. Once I break into a machine, I always get a warm feeling from taking the console log with me as a memento. After all, it is *slightly* possible that someone might read it otherwise. In general leaving an alternate log from another machine running a different operating system on a different style of paper and different type face in its place stops anyone from thinking that anything is wrong.

Get People Upset. While I might not go out of my way to hurt my company, I am sure that a manager with sufficient flexibility could motivate me to do so.

The insider threat can often come from employees just trying to have a good time. In doing this they might accidentally damage your world, or learn skills that can later be used to rip you off. In any event, your audit logs would be contaminated as a result.

> Once upon a time there was a mythical bank. Maintaining the computers there was rather dull. Lots and lots of tape backups to do. Hourly. Mount a tape. Watch it spin. . .
>
> One day a compulsively neat tape mounter was cleaning the place up. He had nothing better to do since Rogue[11] had been removed from the system.
>
> Under a desk he found, taped to the bottom of the center drawer, a password. He and his bored compatriots would randomly transfer various amounts (generally in the hundreds of thousands) between accounts. Just for kicks.
>
> They were astounded that no one every caught on. The password still has not been changed. These operators think that all the transactions have been reversed, but they aren't really sure.

The attack method above is very common.

6 Impersonal Attacks

We all know that fires and floods and earthquakes do indeed happen. And that they happen very rarely. Disk crashes and power failures are much more common.

Keeping two copies of the backup tapes in separate locations, providing that the backups are done frequently enough, is often sufficient unless it is critical that people be able to recover your data quickly at all times. In this case you have the entire array of problems of secondary sites and access methods.

There are companies that sell emergency computer resources. If your computer catches fire or is caught in a flood or earthquake, you bring your data to them and continue to process. A very informal survey of these companies, reveals that this is almost never used.

In practice we find that very often

[11]Rogue is a real time computer game where the player fights his way through a dungeon

- Backups not done regularly enough; in some cases not at all.

  > Once upon a time, there was a small company that sold time on a computer. Due in this case to a malicious cracker (though it could have happened just as easily with an honest programming error) the entire accounting file was lost. The most recent backup was more than a month old and no secondary accounting record was kept. Ten percent of the company's revenue for the year was lost. It is still unclear whether the company will go bankrupt as a result.

- Backups are never checked to see if we can read them.

  > Once upon a time, I saw an entire wall of carefully written tapes that could not be read back. The sight of daily backups gave the system programmers a great deal of confidence until they needed to read one back . . .

- Backups not labeled. Or at least not in any intelligible manner. It should be easy to find which backup tape you want.

- Backups not stored under the right conditions.

- Backups not locked up. Why break into the computer when I can carry a tape home?

More common than fires or floods are sprinkler system failures. And more common than those are disk problems. But much more common than either are programming errors.

If your site can afford it you might consider a disk to disk backup strategy. This is where you have two copies of each file. One on each of two ⇐ different disks. One is "today's," one is "yesterday's". This is *often* sufficient to recover from programmer and many physical errors with no more than a single day's work lost.

Conceptually, you would have twice as many disk drives as you need. Every day at midnight copy the files from one set of drives to the other and then just unplug the second drives until the following midnight. For performance, you instead wish to use half of each disk drive.

Part II
Defenses

Generally computer fences come in two types. The first is the type that stops someone from logging in at all. The second is the type that limits what one can do once he has logged in. I have much more faith in the first type of fence than the second. To draw a weak analogy, it is much easier to prevent all the people from entering a room than to make sure no one breaks anything once they are inside.

7 Passwords: Theory and Practice

The first component of the first fence is the login program. Presuming the login program cannot be tricked, the strength or weakness of the login mechanism ultimately relies on the strength of the passwords. In section 10.1 we will discuss ways to "login" without going through the login procedure.

One person to a password. In addition to giving you security, passwords also give you a level of accountability. With group accounts, it is never clear who either leaked the password or did something that you did not want done. For this reason only one person should know a given password.

Sometimes people believe that several people need to use a single account. I have never seen a case where this is really true. In some cases, people have group accounts where an extra passwording mechanism is added in the `.login` (Unix) or `login.com` (VMS) file. While it is remotely possible to do this in a secure manner, I again have never seen one written that I could not break through.

Once upon a time, a mythical researcher was writing a mythical paper on computer security. In her paper. The portion begging "One person one password" appeared much as it did above. Every one of her pre-readers added a comment of "except for the following special cases." Not a single one of the special cases were vaguely justifiable on any grounds what-so-ever.

No Vendor Supplied Passwords. Everyone and his brother knows the standard passwords that come with a system.

Hidden. No one, including your system manager, should be able to read the unencrypted passwords.

> Once upon a time, there were two computers living about 10 feet from each other. The first was a relatively secure machine. Passwords were required. The logs were monitored carefully. Everything was really done right. Almost anyway.
>
> The second machine was much more relaxed. It was a research machine. There were no file protections. Passwords were required only from dialup lines and distant networks as a token effort to keep off randoms until they learned the rules. Randoms were always given accounts.
>
> Obviously this second machine had no protection on it in any real sense. No one saw a need to protect passwords. In fact, the passwords were stored in plain text in a moderately well hidden file. One day a cracker found the password file and found out the system manager of the first machine had the same passwords on both machines. . .

Had the passwords at least been encrypted on the second machine in our sad tale, the cracker would have done a little bit of work to break into the first machine. Still not too much work.

Hard to Guess. confer figure 11.

Password Aging. This is often required by law. Realize that changing your password ever day will only make it twice as hard to guess as never changing, presuming that no one can see you type it. Password Aging does help to get rid of old, no-longer-used accounts.

Initial Passwords should not be the Same. There is a tendency for system managers to give everyone the same initial password. If he doesn't require them to change it quickly enough it may never change.

> Once upon a time there was a defense contractor with a SECRET Computer. In addition to the obvious problems of the door to both the machine room and the building being left ajar one day, the password for 168 out of the 400 accounts was `Horses` which was posted on the bulletin board of the machine room as the initial password to give everyone.

Initial Passwords should not be predictable.

> Once upon a time there was a university computer science class. All of the accounts had such imaginative names as `cs01-aa`, `cs01-ab` ... `cs01-zz` with equally imaginative passwords as `pwdAA01`, `pwdBB01` . . . `pwdZZ01`. Surprisingly, someone was able to guess the pattern from a single example, causing lots of hassles by systematically *Trojan horsing* almost every student account on the machine before many of the students had logged on. This same student had even more fun guessing the form of the teaching assistant's password.

Initial Passwords should be usable only once. Or they may never get changed and have almost certainly been written down

Too Many Guesses Don't let people get too many guesses in without your noticing. Methods for this are discussed in 2.1. ⟸

Protect The Cipher Text. The encrypted versions of the passwords are useful to the attacker. With them, and either an understanding of the password algorithm or a machine with a similar operating system to yours, he can do his guessing in ways you can't detect. The changes to Unix necessary to move the passwords to a protected place are not very difficult. ⟸

Use a Different Password Cipher Method If you can protect the code that does the password encryption, then even getting the encrypted passwords doesn't help very much. In general this is a hard thing to do.

↩ (The null password.) Very common.

GUEST also ARPA, ANONYMOUS, and RANDOM.

Names Especially Women's Names. SUSAN is a good guess.

Single Characters possibly repeated. Especially 'xxxxxx'.

User Data Any word in the user's identification field. His first name, last name, initials, initials twice.

Company name Or the computer or division name.

User Names User names = password.

ROSEBUD It was used as an example in a manual once. If your manual has an example of setting a password, be sure that that word is disallowed.

Password or passwd for the literal minded.

Short words like LOVE is a good guess. So is almost any piece of profanity.

Classical Passwords like OPEN or SESAME.

Birth date or social security number.

qwerty or other keyboard patterns. Subparts of the alphabet like 'abcdef' have worked nicely.

Backwards Anything above reversed.

...0 any of the above followed by a single digit. Especially, 0, 1, or 9.

Figure 11: Good Guesses at Bad Passwords

Since we have talked about bad passwords, lets take a moment and talk about good passwords. There is a great debate that goes on and on and on and on, ad infinitum, about whether users should be allowed to choose their own passwords. At first this will sound strange but there are really only about 1000 words in Common English. With a thousand guesses you are likely to get a lot of passwords because they tend to pick common English words. In fact they are more likely to pick moderately common English words and that is about four hundred. This means that if you guess a moderately common English word on about four hundred accounts you will probably get one.

Not counting capitalization or non-alphabetic characters, there are about 25 interesting single letter combinations, about 90 interesting two letter combinations, about 760 for three letters 2,150 for four, 3,100 for five and the curve begins to flatten out quickly. A six letter password made up of two randomly selected three letter "words" would have over half a million possibilities and be a little bit difficult to guess. Choosing two four letter combinations would give us over 4 million possibilities. ⟸

Getting people to choose several little pieces to make up a password rather than one big piece would put more randomness in it. Using a system like passphrases[12] is also good if your using community is willing to accept ⟸ them. The pronounceable syllables scheme seems to get the displeasure of users.

Eventually people will have to start picking better passwords or they will be, by necessity, machine generated. Asking people to concatenate two small words seems like the best compromise for the moment. ⟸

In most networking schemes passwords are transmitted in clear text between the machines. Anyone that can listen to the packets can find a lot ⟸ of passwords. In practice this has not been solved.

8 Protection By Policy

There is a real need for perspective here. In some places the small computer has really fallen between the cracks. The large computers are administered

[12]Passphrases are passwords that consist of the first two letters of each of three or four words from an easy to remember phrase.

size	quantity
1	26
2	91
3	759
4	2142
5	3097
6	3796
7	4045
8	3578
Other	10736
Total:	24474

Figure 12: Number of "words" in /usr/dict/words of each size

very carefully. The small pocket calculators are still being carefully monitored because they were once expensive and the $3,000 to $50,000 computer is forgotten about. To quote:

> "If we protected micro-vaxes with modems as well as we protected calculators, we would be doing real well."

Policy contains a lot more than most of us have thought about. Thinking about the issues that should go into your policy is important.

8.1 Identification

How to identify people is an interesting general question. In person we might trust a drivers license or employee ID card. Over the computer or over the phone it is an even more interesting question.

In school's terminal rooms, lab assistants, often known as *terminal watchers* generally ask to see the student's registration card before fixing a forgotten password. Over the phone we might be able to recognize the voice. Perhaps in a computer link we might recognize the word choice or typing mistakes.

> Once upon a time, there was a small high school that rented time from a timesharing company. The had a small terminal and a slow phone connection and paid a small fee for a small amount of computer time.
>
> It came to pass that one of the high school *urchins* became very interested in programming in "C". He liked doing it a lot. So much so that he managed to use up all the time the high school had bought.
>
> He called up the headquarters of the timesharing company (paying for the call as he wasn't a phone Cracker yet) and asked the secretary there what the root password was so that he could install the new system.
>
> Without even dropping a name, he was supplied with the root password. He logged in as root and found the unencrypted password file. Finally, he read the passwords for all the other high schools and programmed to his heart's content.

One of the strategies to help prevent things like this from happening is the idea of calling people back at known telephone numbers. In this way the attacker would have to break into a trusted office in order to fool someone.

> Once upon a time, there was a large laboratory. Thousands of employees used the same very large, somewhat sluggish, computers. The operations manager had decided that to protect the computers from people tricking them as above, he wold require all of the operators to call the users back at a COMPANY number. The operators knew that if they had to dial more than 5 digits it wasn't in the company.
>
> A young Cracker, after three of his impersonation attempts were frustrated, found that the unprotected company cafeteria had a phone that was "safe." It was a company phone that would not let you dial outside the company. It was of course exactly what he needed to let the computer operator call him back on.

One of the easiest things to forge in the entire world is the header on computer mail. In this way one can pretend that he is anyone that he likes.

> Once upon a time, there was a young college cracker. He wanted to be able to program on much larger computers than his university could afford. By sending out mail with forged headers, he pretended to be the account manager on the machine he wanted an account on. He sent this piece of mail with the forged headers to the account manager's assistant and got not only an account but also a full set of manuals.

In somewhat the opposite vein, you should make sure your users know not to ever tell anyone their passwords. This is especially true over the phone.

> Once upon a time from a mythical crowded terminal room, a naive user was having problems with his program. He called up the consulting staff. The consulting staff member couldn't understand the problem by phone and asked for the user's password so the consultant could log in as the user and try to replicate the problem. Thirty-two other people in that tiny crowded terminal room now also knew the password . . .

8.2 Privileges

The number of privileged accounts should be kept to a minimum. It is often the case that bosses are given privileges on the machine that they neither need nor understand as a sign of prestige. This is a problem waiting to happen. In general it doesn't wait too long.

In addition to keeping the number of privileged accounts to a minimum, keeping the level of privilege to a minimum also is important. On VMS there are many separate privilege bits; on Unix, this is generally best accomplished with appropriate use of groups and set group id programs.

8.3 Registration

Information is a very powerful tool. If you have many computers at your site, it would make a lot of sense to know what you have and who runs what. It has unfortunately been the case that so much time was wasted in the middle of an emergency that by the time the right system manager was found, it was much too late.

It has also often been the case that security tools and threats have been made or found and could not be distributed uniformly because no one knew where/what/whom all the computers were. You could distribute a form similar to the one in figure 13 to all of your employees asking that any that run a computer return them. Once you have the information, both physical and electronic mailing lists can be set up.

8.4 Support and Actual Policy

A certain level of administrative support is necessary for proper security. This includes the policy to not give privileges to people that don't need them and the support to do the appropriate registration. Support also includes explicitly setting policies. Beyond this it also means financial support within your organization.

Training. Training comes in two parts. The first part of this is to train the system managers. This training includes not only explicit security but also the best functional training in their jobs. Security problems come not only through carelessness; but, also in the honest attempt

Computer Security Quick Contact Form

For System Mangers and CSSO's

Your Name	
Mail-Stop	
Office Phone	
Home Phone	
Electronic Mail Address	

Person to contact if you can't be found

His/Her Name	
Mail-Stop	
Office Phone	
Electronic Mail Address	

	machine name	**machine type**	**operating system**	**bldg/room**
	sample.ARPA	***vax–780***	***VMS 4.2***	***275-218***
1				
2				
3				
4				
5				
6				

(Please use the back of this sheet for questions or comments.)

Figure 13: Sample Computer Security Registration Form

to give users services that the system programmers/managers do not know how to provide safely.

Second, some level of training is necessary for the user community. This should include things like:

- How to change a password.
- Never to give out a password.
- File Protections.
- Where to report suspicious events.
- `.rhosts`
- Requests for information. What information should a user not give out when someone asks for it.
- How to use groups on their machine.

> Once upon a time, at a mythical government lab, a system administrator got a computer mail message from the another government lab asking for the network topology and a list of what precautions had been taken to safeguard the computers. While the note appeared to come from the directory of another government lab, it seemed suspicious.
>
> The system administrator broke into the other lab's computer to determine who this man really was and asked the system manager there whether her system had been recently compromised and whether the request was authentic.
>
> After being assured by the system manager that the request was authentic and that this machine had never been broken into *ever*, the system administrator called both the director who had requested the information and the manager at home, using the number he had gotten from the protected database on the second machine, to tell them why he had been suspicious of the request.

It is not unreasonable to have a policy about what your users should do when they answer their phones and hear a modem tone. If an attacker is scanning your exchange, (*i.e.*, dialing up each of your phone lines looking for a computer to talk to) many of your users may get such phone calls.

Equipment. There is some equipment that will make it easier to make a computer facility more secure. The administration should supply it as well as the funding to install it properly.

Logging Devices. A printer that will constantly be recording the security information (and other important data). This printer should be in a secure place.

You may also desire to have a second very small computer to accept the logging information and can in fact have such a computer literally call for help when the first computer is having physical or security problems.

Modem. If you have decided that more obscure modems would help your security enough to justify the cost.

Hotline. A single place where anyone at anytime can call if they are experiencing a security problem. Machines that detect problems could themselves call into such a place as well as the users and the system programmers doing so. Hotlines and similar things are *entirely* ⇐ worthless unless competently manned.

Publication. Regularly publishing something for all your system programmers/managers to keep them up to date on security precautions and policies is a good idea. In a large organization this is especially good; much better than having each cluster or machine having to independently invent the same security fixes.

Neighbors. It is a good idea to occasionally, perhaps every other month, encourage the system programmers/managers in your organization to meet with their counterparts in other organizations. Good ideas and tools about computer security in specific and computer management in general can be exchanged.

Legal. Your Legal department can advise you about policy statements, banner wordings and the legalities and perhaps the logistics of phone taps and traces that may become necessary for you to use. Find out about these things before you need to use them.

> Once upon a time, at a mythical university, there was a computer cracker. He used to call up and harass the system manager over the phone. The system manager kept him on the phone for four hours, while he tried to convince the university to trace the call. He was unable to convince them.

Authorization To effectively protect your systems, the system managers and computer security officers need to have a free hand in doing so. They need various explicit permissions.

1. Permission to poke around on networks. To see what machines are on their local nets, including the unregistered machines that your computer security officers don't know exist.
2. To poke around on phone lines. To dial up phone numbers with your organization and find out what computers have phone lines that shouldn't.
3. To be creative and inventive in getting the users to comply with the policies.

> Once upon a time at a mythical university, researchers were forbidden by policy to run programs that were "not documented." Many of these undocumented programs controlled physical devices and the system protections were not sufficient to prevent them from accidently damaging these devices.
>
> One system programmer got annoyed at this happening and wrote a program called `/usr/local/happy`. When the user ran happy, it happily told him that it was deleting all the files that he owned and showed him a listing of the files that it was deleting. It then waited thirty seconds and said: *Only joking. Aren't you Happy! It is against policy to run undocumented programs like this one.*

4. To be able to trace phone calls when necessary.

Connectivity. requirement of reporting of node connectivity and/or periodic (weekly) checking of node connectivity

Goodwill. people doing this and the system managers/users must believe that the management supports the program.

Ex–Employees. Dealing with termination is a hard problem. You can make this easier by having a policy about what to do with the computer accounts of employees when they leave. You should make sure to let the employee get a copy of his personal files[13] so that he doesn't have to break in to get them.

Compliance. A strategy for getting people to actually do the things you want them to do is important. Making it easy for them to do so and letting them know it is important that they do are good first steps.

> I am much more concerned that people do things right than that they tell us about it.

Secretaries. Secretaries always pose a special type of threat. In the computer world, many of them want to learn a lot about the computer; and, to do their job, need to be able to read a lot of different (potentially sensitive) files. It is worth making an extra effort to check that your secretaries have correct file protection on their accounts and to make life as easy for them as possible.

> Once upon a time at a mythical small company, the boss wanted his secretary to be able to work with his files. He wrote for her a special set-uid shell script[14] that would copy a given file to her directory. Unfortunately, using the infamous "csh - -" bug, described in section 15, everyone else could read his files as well.

[13]People *always* keep personal files on the machine, no matter how strongly you word a policy to the contrary.

[14]Set-uid shell scripts are *always* a bad idea!

"Stolen Software" In the real world it is not uncommon for users to somehow get unlicensed (stolen) software. Your system manager should not have to decide what to do about it when the situation arises.

The stolen software is often very useful to the user (otherwise he wouldn't have stolen it) and a potential place for Trojan horses to be.

A special case of stolen software is trade stolen software. It's more than vaguely possible for a user or a system programmer to try to trade some of the proprietary software on your machine to get some from another machine. Alternately, he might be willing to the trade use of his account for use of someone else's account.

Paper Work. A dumb security plan policy is *much* worse than not requiring security plans at all. Requirements in the security plan can waste a great deal of time, alienate the system staff and provide no security at all[15].

Don't Send Passwords By Mail. Remember that mail is guaranteed to be one of the least secure things on your machine. ⟸

> Once upon a time, at a mythical university, there was a mailing list for the people that managed new accounts. Whenever anyone lost their password or wanted a new account, mail would be sent to this list and whoever fixed it would carbon copy the letter telling the person that it was fixed to the whole group. letters like
>
> Dear Sys-person:
>
> I forgot my password. Could you change back to "susan" the way it is on all my other machines
>
> was not uncommon.
>
> Of course it was not long before the crackers started reading these notes . .

[15]I am currently preparing a separate paper on what *really* should be included in a security plan

8.5 Self Awareness

Protect the things you don't want stolen. Keep Multiple copies of things that you don't want destroyed. Of all the files that ⇐ you have on the computer, very few of them are truly critical. If someone destroys `/usr/spool/news/misc/`, it is unlikely to bother you as much as it will if she destroys `/etc/passwd` or worse . ⇐ `/u0/smith/important-data.only-copy`. It is very much worth your effort to know what you really need to protect.

9 Protection by Configuration

Configuring your system correctly is important. Much of the vendor supplied software is somewhat secure-able, but is not secure in it's default configuration. Many of the security problems in a system are created by mis-configuring it. Here is list of things to do to help make sure you system is configured correctly.

Passwords. Change the system supplied passwords and remove the system supplied unpassworded accounts. Set up a good password policy with NO accounts that have the null password.

Paths/Logical Names. Should not included places that are publically writable. "." should not be in the path.

Start Up Files. Should not be publically writable and should specify full pathnames for all the commands that they reference.

Parent Directories. The directories that contain files that need be protected, need to be protected themselves. As do their parents and so on recursively up the hierarchy.

Trusted Sites. Decide which sites you really want to trust. The default in many systems is to trust them all.

Default Protections. Protect the home directories of all users appropriately and set up their initialization so that when they create files, the files will have the proper protections.

Groups. Learn how to use them. Use a group privilege instead of a stronger privilege whenever possible.

SET-UID Shell Scripts. Don't have any.

Auditing. Simple scripts that look for strange things like misprotected directories or extra Set-UID programs.

10 Protection by Programming

Once one is willing to actually program, the level of defense can be increased a great deal. You can do even more if you can get the source code for the operating system that you are using.

First of all, for your version of the operating system, you can get a list of the known bugs and fixes so that you can install them.

You can move the password or change the password algorithm. You can install a better password changer that stops people from having bad passwords.

You can filter the message facility. It is possible for an attacker to write to the user's terminal in such a way that it doesn't display the information but rather echoes it back to the computer, appearing as if the user himself had typed it.

If you are very ambitious, you can write a program to look at your console logs to find interesting things to report to you.

You can write a facility that will let you know when any of the system binaries are changed. Beyond just looking at modification dates, you can keep track of the particulars *inodes* as well as keeping checksums of the files.

You can install facilities to allow your computer to better record break-in attempts and to inform you (if you are logged on) about them in realtime.

You can write the programs that do things safely rather than being forced to used gross kludges like Set-UID shell scripts.

10.1 Ways to Login Without Going Near the Login Program

Presuming that you have good password policies and proper configurations of trusted sites, the dedicated attacker will have to resort to other manners of entry. Generally this involves tricking daemons[16]. There are five main strategies for this.

Meta Chars. The idea here is to give the daemon a command string that will do something extra when it is used. Consider a Finger Daemon[17].

The user on machine-a might type:

```
finger russell@machine-b
```

Machine-a would then talk to machine-b and machine-b would execute:

```
finger russell
```

locally. This works well and good. However the user on machine-a could just as easily have typed on a single line:

```
finger \;/bin/echo RB::0:0:new account:/:/bin/csh
>> /etc/passwd\;@machine-b
```

which would be executed by the daemon on machine-b as the two commands:

```
finger;
/bin/echo RB::0:0:new account:/:/bin/csh >>
/etc/passwd;
```

The latter creates an account for the attacker named RB with all the privileges on the machine.

It is important to makes sure that daemons run with the minimum ~~privilege they need to get~~ their jobs done and that they check for the ⇐

[16]A daemon is a program, often called a server, that responds to requests made by users. These users, often called clients, may reside on other machines.

[17]Finger is a program that reports who is using a computer and gives information about them.

all meta-character *carefully*.

Intermediate States. It is sometimes possible to start a transaction with a daemon and then crash the daemon in such a way as to leave the system in a mess. Sometimes the mess is useful to an attacker.

New Use. This is the hardest one to anticipate. The attacker can sometimes come up with a new use for the daemons that you had just not guessed at. Things of this nature include using the file transfer facility to get a copy of the core image of the running system or to take a copy of the password file. Debugging facilities tend to provide these new uses even more often then you would think.

Trojan Horses. Very often the daemons come with built in Trojan Horses. The best known example of this is the wiz password to sendmail. In some configuration it has a password on it (which is known to most well-informed attackers even if it is not know to you). The details of this particular Trojan Horse are discussed in section 15.

Bait and Switch. This is where the attacker cleverly kills off the standard daemon and starts up one of his own. He can then of course trick users into supplying passwords and other useful information that he can record.

Part III

Special Types of Sites

11 Special Concerns for Schools

For the Computers that are located in schools there are some special concerns.

Grades. There should be no access by students to the computers that the grades are stored on.

Graded Projects. When students feel that they are competing with one another there is a temptation for them to destroy each other's work. Non-competitive grading is a good idea here. Teaching the students to use file protection is also a good thing. (Self Defense).

Exams. Exams should not be stored on the same computers the students are using. If this can't be avoided, at least encrypt them and give them non-obvious names.

College Hackers. You are likely to have them. Independent projects for credit is a good way to have them do more productive things than harming your system.

Networks. If you are on a computer network, there may be some extra break in motivation to get access to these networks or by the outside world through these networks.

12 Special Concerns for Businesses

The big thing here is financial loss. Understanding what and where your potential losses are is *necessary* for appropriate protection.

Often Businesses are just too busy with business to take appropriate precautions that would protect their machines. Further, they are often using operating systems and standard packages with well known security holes. $\Longleftarrow$

13 Special Concerns for Governments

In a government installation you have a number of special concerns.

Stupid Rules. Often the legal and policy constraints absorb all of the time that might profitably be used protecting the systems rather than filling out forms. $\Longleftarrow$

Compromise. You may have classified, sensitive, or embarrassing information on your computers. These computers should have *no* access from the outside world.

Embarrassment. If your machines are broken into, you may waste more money in dealing with the investigations and the media than you would have spent in establishing good security practices.

Loss of Information. In some case you have the only copy of information in the entire world. You should immediately make sure that there are multiple copies of it within your facility.

Networks. Your machine is probably accessible through a variety of networks. You should find out what they are and what threats they pose.

Part IV

Special Things

14 VMS

If you are running a VMS system there are some special precautions that you should take. You should also realize that many UNIX programs, (especially networking code), have been ported complete with security bugs within it.

Passwords Make sure that none of the user id / password combinations (in figure 14 work. Further, make sure that none of the system supplied accounts have the null password

SYSTEST. Consider setting `DISUSER` for `SYSTEST`.

File Read Protection. Ensure that the following files are protected against any sort of access by the world. Remember `WORLD` probably includes more people that you think it does if you are on any sort of ⟸
a network at all.

Remember that if you edit a file, the new file does not inherit the protection of the old one. ⟸

name	password
system	manager
field	service
guest	guest
play	games
games	play
systest	uetp

Figure 14: Bad Choices for VMS Passwords

SYSUAF.DAT. Read access will give away privileged usernames for the use as targets for password guessing. Further, the attacker can bring the encrypted passwords back to his own machine to guess at his leisure.

Listings from AUTHORIZE. Again we have the target names for password guessing.

Pagefiles, Swapfiles, Dumpfiles They potentially contain unhashed passwords.

AUTHORIZE.EXE. Normal users have no need to use this. Your site may have other authorization programs that might also be useful to crackers.

File Write Protection. Make sure that all the binarys and `.com` files on the system, as well as their directories, are not publically overwritable. This should be true of all of the user's home directories and executable files.

Decnet. Remember that with the normal set up of proxy logins `WORLD` includes anyone on your local decnet.

Spawn. If you expect that certain accounts (perhaps GUEST) can only use a limited number of programs (perhaps MAIL), you should check that they cannot spawn new `DCL`'s. There is a restriction that you can set up with `AUTHORIZE` to prevent this.

tcp/ip. If you are running TCP/IP as part of your networking code, check that you don't have `FTP` running with `SETPRIV`.

Understand the Privileges. Any one of `BYPASS`, `CMEXEC`, `CMKRNL`, `DETACH`, `LOGIO`, `OPER`, `PFNAM`, `PHYIO`, `SYSNAM`, `SETPRV`, `SYSPRV` or `VOLPRO` is probably sufficient for the average attacker to take control of your system.

Any one of `ALTPRI`, `BUGCHK`, `EXQUOTA`, `GRPNAM`, `PRMGBL`, `PRMMBX`, `PSWAPM`, `SYSLCK` or `SYSGBL` is probably enough to allow the attacker to force you to reboot the machine. Your system manuals will have a fuller explanation of this.

15 UNIX

Several complete books could be dedicated to the known security holes in UNIX. In general they are caused by:

- Not using the file protections correctly.
- Running things Set-UID that should not be.
- Meta-characters (discussed in section 10.1).
- Pre-installed bugs.

Let us now look at some selected examples of security problems with Unix.

Sendmail. Some version of the mailer come with a Wizard password. In your `sendmail.cf` file the encrypted form of this password is on the line beginning with the letter `O`. It allows anyone that knows the password to become root on your machine from any machine on the same network as yours. You can compile your `sendmail.cf` into a `sendmail.fc`, which is known as a frozen configure file, and in doing so, set the password to be the empty string.

In some versions of sendmail there is a `-C` option that allows the user to specify a different configuration file. If this file is misconfigured it is displayed for the user without regard for its file protection.

Crontab. Often `/usr/lib/crontab` is underprotected. By writing into it, anyone can get an arbitrary program to run as root.

csh - -. On many versions of `csh`, you can take any Set-UID shell script and ask it to run an alternate shell script (named "`-`") with its privileges. Having no Set-UID shell scripts or fixing the problem in `csh` are solutions to this threat.

Stolen Password Files. Since people can see the encrypted passwords, they can guess them undetected at their leisure. The password field should be moved out of `/etc/password` and protected. ⟸

Mailing into Password File. Sometimes the mailer is running with sufficient privileges that someone can mail an entry unto the end of `/etc/passwd` or some other critical file.

Set-UID. Using Set Group ID programs instead of SET-UID programs can make unix more secure.

core. Often programs that have unencrypted data in them can be aborted and will dump core containing the data that you wish to protect. Setting all your binaries to be executable, but *not* readable, solves this and several related problems.

kmem. Failing to protect `/dev/kmem` and similar files allows the attacker to watch every character that every user types. Setting this device to be readable only by a group and setting the Set-Group-ID bit for programs that need to look at it is a good solution.

ftp. The file transfer program can allow people to read any world readable file without identifying themselves. You can restrict ftp to allow only certain parts of the file system to be accessed.

tftp. An alternate version of ftp. It has a separate server and if you restrict ftp, remember to restrict tftp as well. This you must do by altering the source code.

Group 10. On many Unixes, the world is set up for group 10 to be the staff group. Don't put everyone in it. Similarly for group 0.

Paths. Don't put "." in a path. Use full path names everywhere. Including in rc files. Make sure that nothing in the standard path is world writable.

.rhosts hosts.equiv Restrict their use and/or educate your community.

rdis. The remote distribution mechanism that often allows an attacker to get an entire cluster of machines rather than just one.

popen. This system call tends to lead to insecure code with the meta character problem discussed in section 10.1.

expreserve. This facility saves editor files when `vi` is abnormally terminated. By altering the `IFS` (interfield separator) one can subvert this daemon. Altering the source code is necessary to fix it.

adb. In some Unixes one can interrupt Set-UID or SET-GROUP-ID programs, alter them and then continue them with their former privileges. If you make the binaries execute only, this problem is prevented.

ptrace. In some Unixes one can modify the shared image in memory of a running program. If you make the binaries execute only, this problem is prevented.

16 Tiger Teams

Tiger teams are a lot of fun. This is where you tell some people to go and try to break into your system. *If* you can convince your management to allow this, it is a good thing to do.

A variation on the this theme is role playing. This is where people pretend to be doing the tiger strikes but instead do it all hypothetically.

While it is not as much fun as a Tiger team, one can learn a lot if several different computer facilities are represented.

17 Trojan Horses

Your system probably contains a numer of Trojan Horses. Some of these came with your system like the wizard password in section 15. Others have probably been made by your users.

Anytime one runs code that he has not himself written and has not been audited, one runs the risks of being Trojan Horsed.

By selecting what software is installed on the machine and protecting the appropriate directories and paths, one can minimize the risk of Trojan horses.

18 Conclusions

There has been a lot of detail to wade through so far. In fact, you can get lists and lists of bugs and attack histories and maybe do something useful with them. Since human beings can remember *only seven* facts, the entire content of this session may be found in figure 15.

A A short *History*

In the beginning, computer security was achieved through information hiding; the computers were sufficiently difficult to use that additional security was unneeded. This approach, called *Security through obscurity*, is the basis of several University systems, including the Incompatible Timesharing System (ITS[18]).

On the following *day*, we determined that secretaries and administrators and other non-hackers should be able to use the computer system. Clearly, if we made the system simple enough for them to use, we would no longer have enough inherent obscurity. We therefore packed all the obscurity into

[18]Developed at the MIT AI lab.

One. Know what you want. Say it in your policy. Tell it to your users and your administrators.

Two. Know what you've got. Network access and topology. Who your users are. What the common threats to your operating system are. Part of understanding the threat is to know what is worth stealing or destroying on your system and especially protecting these things.

Three. Watch what happens. An unwatched Console log never boils.

Four. Don't let the users choose between security and Convenience. You know how they will choose.

Five. Use Good Passwords. Always.

Six. Install it right. The first time, each time and recheck it occasionally.

Seven. Technology and Policy must agree. If your system programmers want to administer one policy and you another, you are guaranteed to get neither. The system programmer will come closer to winner than the administrator. Always.

Eight. Test it. Never just trust anything that you can test instead.

Figure 15: The Seven Golden Rules

a word or two and called it a password. We let each user choose their own password so that it would be obvious to them and obscure to everyone else.

On the third *day*, we found out that there wasn't very much obscurity (which we began to call randomness after reading Shannon's work) in the passwords that the users selected. They chose their spouse's name or perhaps their own. It was pretty easy to guess. As a result we picked passwords for them. These were known to be sufficiently random. So random that the people would either forget them, or write them down. Often both.

On the fourth *day*, we found that people told their passwords to each other. This was so they could get to each other's files. We invented a password scheme such that only one session could use the password at a time to discourage this.

On the fifth *day*, we saw that people would walk away from their logged-in terminals. The system response time was sufficiently slow and logging in so difficult that this made sense. We developed gunner programs that would detach their jobs from the terminals after a certain amount of idle time without loosing the state. We gave to the user facilities to detach in this manner at will and to lock their keyboards.

It is now the sixth *day*. The day we solve the week's problems. This is the day that we learn to use the full power of cryptographic number theory, the day that we give users passwords that they can remember, the day that we learn to identify and authenticate the user on each command without bothering or distracting him, so as not to need passwords at all.

And on the seventh *day*, we will rest in our glory until the new breed of computer Cracker is born. I expect it will be a very short day.